ARQUITECTURA DE SOFTWARE PARA LOS ACTUALES SISTEMAS CIBER-FÍSICOS

SOFTWARE ARCHITECTURE FOR THE CURRENT CYBER-PHYSICAL SYSTEMS

ARCHITECTURE DES LOGICIELS POUR LES SYSTEMES CYBER-PHYSIQUES ACTUELS

Jeannette S. Ting
University of Southern California
jeannette.ting@usc.edu

(Tipo de artículo: **INVESTIGACIÓN**. Recibido el 10/02/2011. Aprobado el 15/04/2011)

RESUMEN
La próxima generación de sistemas ciber-físicos –CPS– plantea grandes desafíos en el diseño de software. No se trata sólo de diseñar un sistema en torno a los plazos de ejecución, lo más importante es maximizar la utilización de recursos. En este artículo se propone un sistema de base para la arquitectura de software en el que los servicios se puedan diseñar e implementar y componer fácilmente de acuerdo con la demanda y mediante aplicaciones individuales, de manera que satisfagan requisitos específicos de confianza, seguridad, eficiencia, confiabilidad y previsibilidad, mientras que permanecen dentro de la límites de las capacidades del hardware determinado

Palabras clave
Arquitectura de software, Sistemas ciber-físicos, arquitectura de hardware.

ABSTRACT
The next generation of cyber-physical systems -CPS- poses great challenges for software design. This is not only about designing a system around execution times; the most important issue is to maximize resource's utilization. In this article we propose a base system for software architecture having services that can be designed, implemented and easily composed according to demand and by means of individual applications, in such a way that satisfy specific requirements of trust, security, efficiency, reliability and predictability, while staying within the limits of the capabilities of specific hardware.

Keywords
Software architecture, Cyber-physical systems, Hardware architecture.

RÉSUMÉ
La prochaine génération des systèmes cyber-physiques –CPS- pose des grands défis dans la conception de logiciels. Ce n'est pas seulement au sujet de concevoir autour des délais d'exécution, le plus important est maximiser l'utilisation des ressources. Dans cet article on propose un système de base pour la architecture des logiciels pour laquelle les services peuvent être conçus, implémentés et composés d'une manière facile conformément à la demande et grâce aux des applications individuelles, de sorte qu'ils satisfassent des conditions spécifiques de confiance, sécurité, efficacité, fiabilité et prévisibilité, pendant que ils restent dans les limites des capacités du hardware particulier.

Mots-clés
Architecture des logiciels, Systèmes cyber-physiques, Architecture de hardware.

1. INTRODUCCIÓN

La próxima generación de sistemas ciber-físicos –CPS– plantea grandes desafíos en el diseño de software debido a factores como: 1) la inmensa diversidad de plataformas hardware, sobre las que se implementan las aplicaciones distribuidas e integradas; y 2) la diversidad de aplicaciones es sí mismas y sus requisitos. Los requisitos de las aplicaciones no sólo tienen restricciones en tiempo real, sino que también en factores como seguridad, fiabilidad y confianza. Ya no es simplemente diseñar un sistema en torno a los plazos de ejecución del peor caso, por lo que lo más importante es la maximizar la utilización de recursos. Mientras que estas consideraciones serán significativas, el foco principal del diseño del sistema será el de asegurar un comportamiento confiable, seguro, eficiente y previsible de las aplicaciones compatibles [1]. Esto es evidente dado el continuo aumento en la complejidad del software, con decenas de millones de líneas de código previstas para un futuro cercano en dispositivos embebidos, que van desde los teléfonos móviles a los sistemas del automóvil [2].

La diversidad de plataformas de hardware tendrá un gran impacto en el diseño del software, desde las abstracciones de servicios básicas hasta la programación, la comunicación y coordinación entre los servicios y aplicaciones. Además, el uso juicioso de las características de hardware será esencial para garantizar el uso de eficiente de recursos –por ejemplo, memoria, batería, ciclos de CPU y ancho de banda de comunicación–, así como garantizar los niveles necesarios de fiabilidad, seguridad y confianza.

Lamentablemente, muchos de las actuales investigaciones en sistemas se centran en la ingeniería y la extensibilidad de los sistemas comerciales disponibles en el mercado (COTS), para cerrar la brecha semántica entre las necesidades de cada aplicación y las disposiciones de servicio del sistema. Se dedica un esfuerzo significativo para eludir las limitaciones de varias estructuras de sistemas, los API y servicios genéricos se diseñan de manera independiente al uso. Por ejemplo, los sistemas monolíticos como Linux se pueden extender para que los usuarios sin mucha experiencia incorporen sus propias políticas de servicio –por ejemplo, que consideren explícitamente los plazos de solicitud para programar la CPU– al dominio de la protección privilegiada del núcleo central [3]. Sin embargo, esto puede comprometer la integridad y, por tanto, el comportamiento tanto del propio sistema como de las aplicaciones residentes. Sin embargo, en consecuencia, se ha invertido un gran esfuerzo para tratar las operaciones con funciones de memoria usando características de hardware como la paginación y la segmentación, y las técnicas de software, tales como lenguaje del tipo seguro y el aislamiento de errores basados en software. Otras estructuras del sistema, tales como micro-núcleos, servicios específicos de aplicación aislados del núcleo central, aunque mucho esfuerzo de investigación en el diseño de estos sistemas se centra en técnicas eficientes para reducir los costos de comunicación inherentes impuestos por servicios aislados. Fundamentalmente, un significativo cuerpo de trabajo ha tratado de abordar las deficiencias de varios diseños del sistema debido a la falta de correspondencia entre sus suministros de servicios y las necesidades de la aplicación [4].

Teniendo en cuenta los diferentes requisitos de aplicación y las características de hardware de los futuros CPS, no será suficiente una solución de diseño para el sistema. Más bien, lo que se requiere es un sistema de base de arquitectura de software en el que los servicios se puedan diseñar e implementar y componer fácilmente de acuerdo con la demanda y mediante aplicaciones individuales, de manera que satisfaga requisitos específicos de confianza, seguridad, eficiencia, confiabilidad y previsibilidad, mientras que aún permanecen dentro de la límites de las capacidades del hardware determinado.

2. EL FUTURO SOFTWARE DE SOPORTE PARA LOS SISTEMAS CIBER-FÍSICOS

La visión es diseñar un sistema como una colección de servicios y abstracciones de aplicación específica, que sean estructurados y desarrollados automáticamente en una plataforma destino de acuerdo con las limitaciones en términos de: a) las capacidades del hardware, y b) los requisitos de aplicación. Esto difiere desde el punto de vista de que una estructura sin lugar a dudas siempre es superior a otra.

Más bien, esta visión reconoce que la estructura óptima del sistema depende de las características de hardware y las necesidades de aplicación. Por otra parte, el sistema debe estar integrado automáticamente sólo con los servicios necesarios para la tarea en cuestión. Por ejemplo, características como un subsistema de administración basado en discos podría ser pertinente para una aplicación embebida –como en la aviación, los sistemas automovilísticos y aplicaciones futuras en el hogar– [5]. En los sistemas tradicionales, la carga de servicios innecesarios ha impactado recursos como la memoria y en la complejidad del código. Las consecuencias de eso pueden dar lugar a violaciones de seguridad y de previsibilidad, lo que hace razonar acerca del comportamiento de un sistema intratable.

2.1 Superar los vacíos en la Semántica

En los sistemas actuales, las aplicaciones solicitan servicios al núcleo de confianza a través de una interfaz bien definida –por ejemplo, mediante llamadas al sistema. Estas interfaces o APIs, generalmente se definen en un nivel que satisfaga las necesidades comunes de un amplio espectro de aplicaciones, lo que hace que sea incómodo para las aplicaciones especificar exactamente qué comportamiento necesitan del sistema subyacente. Los futuros sistemas ciber-físicos deben permitir que las aplicaciones sean programadas utilizando la interfaz del sistema más adecuado. Al permitir que las

aplicaciones especifiquen precisamente su servicio y necesidades de recursos, el sistema podrá de forma transparente auto-organizarse en la estructura más beneficiosa, utilizando los servicios más apropiados. Por ejemplo, un cuidadoso diseño API puede permitir que una aplicación de aviación pueda especificar las solicitudes de servicio para la altitud y el control de velocidad, de modo que se alcancen la altura y los requisitos específicos de velocidad de aire [6]. De forma clara, el sistema compone de una serie de servicios de nivel inferior –por ejemplo, influir en el timón, el alerón y el comportamiento del motor–, que se organizan de acuerdo con los requisitos de aplicación y las características de hardware.

2.2 Una propuesta para la arquitectura de software

Los puntos de vista antes mencionados sugieren un sistema que debe estar estructurado como un colección de servicios de componentes, que pueden ser aislados –utilizando técnicas de hardware y/o de software– o combinados en un sólo espacio de dirección de acuerdo con requisitos bien definidos. Por ejemplo, si el proceso subyacente del hardware tiene una unidad de gestión de memoria –MMU–, una serie de servicios de componente podrá automáticamente asignarse para separar los dominios de protección del hardware, cuando este grado de seguridad sea un requisito importante de la aplicación [7]. Este aislamiento incurre en demoras entre los distintos servicios de comunicación, por lo que el sistema automáticamente debe arreglarlo con los canales de comunicación creados dinámicamente para satisfacer el equilibrio entre los requisitos de latencia y el aislamiento.

Del mismo modo, temas como la previsibilidad, con la que las invocaciones de servicios se producen en puntos bien definidos en el tiempo, puede ser considerada por el sistema ya que se adapta a la mejor organización posible. En esencia, una configuración de sistema pueden parecerse a un micro núcleo, un monolito, una jerarquía de servicio por capas, una máquina virtual distribuida, un único espacio compartido de direcciones con tareas de la aplicación, o una combinación en función de las capacidades del hardware y los requisitos de aplicación. El desafío es obtener la mejor configuración desde una serie de descripciones de servicios de alto nivel escritas por los desarrolladores del sistema de forma independiente al hardware, por lo que los servicios de componentes cuidadosamente seleccionados para la aplicación específica son automáticamente dirigidos a una plataforma de hardware determinada. En efecto, el desarrollador del sistema ya no tiene que preocuparse por el diseño del sistema de una plataforma de hardware específica, sólo debe centrarse en los servicios disponibles para las aplicaciones [8].

Una arquitectura de sistema de ciber-físicos puede estar integrada por un ejecutable pequeño capaz de componer la dinámica de servicios y aislar los servicios de los componentes. Este ejecutable estará facultado con características capaces de localizar, autenticar, recuperar y comunicarse con servicios remotos, ya que es poco probable que todos los servicios para una aplicación distribuida embebida estén locales físicamente. Los protocolos de comunicación adecuadamente diseñados serán necesarios para que los servicios desplegados en potenciales entornos de área amplia puedan ser accedidos e invocados por los clientes de confianza, de acuerdo con las limitaciones de ancho de banda y el retardo, así como en seguridad.

Debido a la heterogeneidad del hardware, un sistema cibernético-físico puede abarcar múltiples dispositivos con diferentes arquitecturas de conjunto de instrucciones –ISAs. Tradicionalmente, la invocación de servicios ha involucrado complejos procedimientos de cálculo de referencias para el intercambio de datos entre plataformas de manera independiente de la arquitectura. Teniendo presente además, que para los servicios distribuidos es deseable un sistema unificado de ISA independiente del hardware. Cuando un servicio es implementado sobre una plataforma destino es recompilado y posiblemente vinculado con la verificación, la comunicación, la protección y otro código de contenido para un conjunto de instrucciones de hardware específico. Aunque esto es similar a la idea detrás del código-byte de Java, mecanismos como las máquinas virtuales de tiempo no real, la compilación justo a tiempo –JIT–, y los tipos de seguridad forzados por software no son apropiados en sistemas embebidos con recursos limitados y limitaciones de predictibilidad. Dado que los servicios se compilan en un dispositivo dependiente del ISA en demanda, son posibles las optimizaciones que satisfagan las limitaciones tanto del hardware como de la aplicación. Por ejemplo, la función *.inlining*. puede aplicarse a los dispositivos de memoria, pero no necesariamente en aquellos con memoria limitada. Mientras los desarrolladores de sistemas producen servicios compilados para un hardware ISA independiente unificado ISA, los dispositivos embebidos por sí mismos pueden tener las capacidades para compilar localmente los servicios para su arquitectura específica. Del mismo modo, la compilación cruzada sobre servidores relativamente potentes puede ser adecuada para los servicios asociados con dispositivos que tienen insuficientes recursos para ejecutar sus propios compiladores fuente [9].

3. CONCLUSIONES

Las principales limitaciones de los sistemas ciber-físicos de hoy, desde una perspectiva del software de sistemas, son las siguientes: 1) hay una inconsistencia entre las necesidades de aplicación y los aportes de servicio del sistema debido a APIs inadecuadas, lo que hace dificulta la posibilidad que las aplicaciones especifiquen con precisión el servicio que desean; 2) los servicios del sistema tienden a ser aplicaciones agnósticas, ya que por lo general se centra en la equidad y la eficiencia en lugar del tiempo, la confianza, la seguridad, la fiabilidad y las limitaciones;

y 3) los sistemas actuales son poco flexibles para ser fácilmente extendidos con servicios específicos para aplicaciones destino.

Los desafíos que enfrentan los sistemas de apoyo para CPS futuros incluyen: 1) el diseño de una arquitectura de software subyacente que se organice así misma con los métodos más adecuados de comunicación y el aislamiento entre servicios; 2) la composición automática de servicios para satisfacer las limitaciones de la aplicación, teniendo en cuenta las limitaciones del hardware subyacente; y 3) el diseño cuidadoso de APIs y la consideración de la heterogeneidad de hardware en la generación y verificación de un sistema software para una aplicación dada.

REFERENCIAS

[1] Y. Agarwal, T. Weng & R. Gupta. "The Energy Dashboard: Improving the Visibility of Energy Consumption at a Campus-Wide Scale". *In ACM BuildSys '09*, California, USA, Nov 03. 2009.

[2] X. Fan, W.-D. Weber & L. A. Barrosso. "Power provisioning for a warehouse-sized computer". *ACM SIGARCH Computer Architecture News*, Vol. 35, No. 2, pp. 13-23. 2007.

[3] O. Avissar, R. Barua & D. Stewart. "An optimal memory allocation scheme for scratch-pad-based embedded systems". *Trans. on Embedded Computing Sys.*, Vol. 1, No. 1, pp. 6-26. 2002.

[4] T. A. Henzinger. "The theory of hybrid autómata". In M. Inan & R. Kurshan, editors, Verification of Digital and Hybrid Systems, volume 170 of NATO ASI Series F: Computer and Systems Sciences, pp. 265-292. London: Springer-Verlag. 2000.

[5] H. Sutter & J. Larus. "Software and the concurrency revolution". *ACM Queue*, Vol. 3, No. 7, pp. 54-62, September 2005.

[6] E. A. Lee. "The problem with threads". *Computer*, Vol. 39, No. 5, pp. 33-42. 2006.

[7] M. Lave & J. Kleissl. "Maps of optimal tilt and azimuth for PV systems and advantages of tracking over the U.S". *International fair for renewable energy Solar Energy 2010*, Berlín, Germany, Feb. 16-20. 2010.

[8] E. A. Lee. "Model-driven development - from object-oriented design to actororiented design". *Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation* (a.k.a. The Monterey Workshop), Chicago, USA. Sept. 24. 2003.

[9] S. Johannessen. "Time synchronization in a local area network". *IEEE Control Systems Magazine*, Vol. 24, No.2, pp. 61-69. 2004.